|  |  |
| --- | --- |
| **面向对象程序设计 综合训练报告** | |
| 图片包含 标牌  已生成极高可信度的说明 | |
|  | |
|  | |
| **姓名** |  |
| **班级** |  |
| **学号** |  |
| **电话** |  |
| **Email** |  |
| **日期** |  |

# Minesweeper Game

题目 1数据设计 3

算法设计 4

主干代码说明 5

运行结果展示 8

总结和收获 9

源代码 10

# 题目

# 

# 数据设计

为实现本题所述的扫雷游戏，设计了以下6个类：

1. MineSweeperMain:这个类是扫雷游戏的主程序，定义了扫雷游戏的主面板，其中包括了菜单面板、游戏面板和新游戏面板。该类继承了类JFrame，并需包含菜单面板、游戏面板和新游戏面板三部分，即类GameMenuPanel，NewGamePanel和GameBoardPanel；
2. GameMenuPanel:这个类是扫雷游戏上方的菜单栏、进度条和计时器部分。该类继承了类JPanel，且其需包含其上部的菜单栏JMenuBar，菜单栏中的菜单JMenu，菜单中的三个菜单项JMenuItem，同时还需包含下部的进度条和计时器栏JPanel，进度条JTextField，计时器JTextField。为实现计时功能，需要存储开始游戏的时间，并利用一个每1秒自动运行的Timer更新计时器示数；
3. NewGamePanel:这个类是扫雷游戏下方的开始游戏区域。这个类继承了类JPanel，并同时实现了ActionListener接口，以对开始新游戏的行为进行监听处理。这个类中包含四个JButton，即开始游戏按钮和三个选择难度按钮，同时还定义了定值EASY,INTERMEDIATE,DIFFICULT以便向其他类传参；
4. GameBoardPanel:这是扫雷游戏的主游戏面板。类中包括了一个Cell类的二维数组、雷区的大小（行数、列数、雷的大小）和地雷的个数。类中还定义了一个内部类继承了MouseAdapter，用于对雷区内的鼠标行为进行处理；
5. Cell:这是主游戏面板类所用到的类，代表其中的每一个块。其中存储了块的位置（行、列）和状态（是否被挖、有地雷、被插旗），还包括块的绘制信息；
6. MineMap:这也是主游戏面板类所用到的类。它代表整张扫雷地图，利用一个boolean二维数组存储地图的含雷信息；

最后，为便于各个面板间的相互操作，在GameMenuPanel、NewGamePanel和GameBoardPanel中需包含一个类MineSweeperMain的引用，并在对应的对象构造时将当前的主面板传入。

# 算法设计

1. MineSweeperMain类是扫雷游戏的主程序，需要包含main方法并在其中安全地启动构造方法。在构造方法中，初始化这个JFrame；
2. GameMenuPanel类涉及的方法有：
   1. 构造方法：初始化这个JPanel以及其中的JMenu等，并为菜单项添加鼠标监听器；
   2. refreshBar:调用此方法时，刷新进度条。采取遍历全图的方式得到未发现的地雷个数，并更新JTextField中的内容；创建新游戏、重置游戏、挖雷或插旗后，都要调用此方法；
   3. startClock:调用此方法时，开启计时器。获取和存储当前时间，并设置一个新Timer使之每1秒更新一次JTextField中的内容；创建新游戏或重置游戏时，要调用此方法；
3. NewGamePanel类涉及的方法有：
   1. 构造方法:初始化并添加动作监听器；
   2. actionPerformed:这个类本身实现了ActionListener接口，可以处理开始新游戏的行为，因此其他类中的开始游戏按钮也可利用它为监听器；接收到开始游戏相关的行为后，清空JPanel，添加三个难度按钮供用户选择；接收到难度按钮的行为后，恢复JPanel，以对应难度调用GameBoardPanel里的newGame方法；
4. GameBoardPanel类涉及的方法有：
   1. 构造方法：避免每次开始新游戏都要创建新的GameBoardPanel对象，因此将该类的初始化行为合并到newGame方法中；
   2. newGame:根据难度参数，设定ROWS,COLS,numMines；创建新的cell二维数组并为每一个Cell添加监听器（内部类的对象）；
   3. resetGame:重置游戏，将每一个Cell的被挖状态、插旗状态复位并重绘即可；
   4. getSurroundingMines:采用遍历返回当前格上应显示的数字；
   5. revealCell:挖雷操作，采用遍历和递归，在当前格为0时对周边也进行挖雷；
   6. hasWon:两次遍历cells判断两种获胜条件，第一种不满足的情况下开始判断第二个；
   7. 内部类CellMouseListener的方法mouseClicked:若左键，判断是否为地雷，若是玩家输，否则调用revealCell；若右键，判断应插上还是移除旗子，更改Cell的isFlagged属性并重绘；此后，判断玩家是否赢，玩家赢或输时，停止计时器并利用JOptionPane.showMessageDialog弹出提示；
5. Cell类涉及构造方法、初始化方法和绘制方法；
6. MineMap类只涉及构造方法，其中利用一个Random对象生成随机数，作为MineMap里的地雷位置；

# 主干代码说明

以上是所涉及主要类之间的UML关系图；

**public void** refreshBar() {  
 **int** notFoundMines=**frame**.**board**.**numMines**;  
 **for**(**int** row = 0; row < GameBoardPanel.*ROWS*; ++row){  
 **for**(**int** col = 0; col < GameBoardPanel.*COLS*; ++col){  
 **if**(**frame**.**board**.**cells**[row][col].**isFlagged**){  
 notFoundMines--;  
 }  
 }  
 }  
 String text = **"剩余地雷数："** + notFoundMines + **"/"** + **frame**.**board**.**numMines**;  
 **if**(notFoundMines>0){  
 text+=**"⬛"**.repeat(notFoundMines);  
 }  
 **progressBar**.setText(text);  
}

以上是实现刷新剩余地雷数进度条的代码；

**public void** startClock(){  
 **timeBar**.setText(**"00:00"**);  
 **timeStart**=System.*currentTimeMillis*();  
 **timer** = **new** Timer(1000, e -> {  
 **long** timePeriod=System.*currentTimeMillis*()-**timeStart**;  
 **timeBar**.setText(String.*format*(**"%02d:%02d"**,timePeriod/60000,(timePeriod/1000)%60));  
 });  
 **timer**.start();  
}

以上是实现开始计时的代码；

@Override  
**public void** actionPerformed(ActionEvent e) {  
 removeAll();  
 **if**(e.getSource()==**btnNewGame**||e.getSource()==**frame**.**menu**.**menuItemNewGame**){*//单击新游戏按钮时，让用户选择游戏难度* add(**btnEasy**);  
 add(**btnIntermediate**);  
 add(**btnDifficult**);  
 updateUI();  
 }  
 **else**{*//单击难度按钮时，以对应难度开始游戏* add(**btnNewGame**);  
 updateUI();  
 **if**(e.getSource()==**btnEasy**){  
 **frame**.**board**.newGame(***EASY***);  
 }

以上是NewGamePanel类实现ActionListener接口作为开始游戏相关事件监听器的主干代码，该段代码实现了接口中的actionPerformed方法，这个类也被复用于菜单中的开始游戏菜单项的监听；

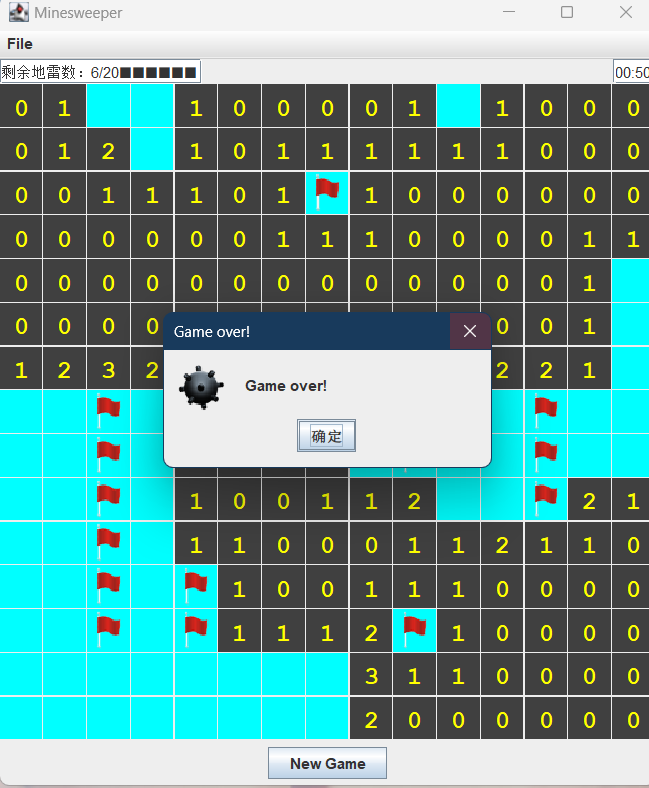
**public boolean** hasWon(){  
 *// 获胜条件1：所有单元格都成功执行了挖雷操作  
 // 获胜条件2：所有的地雷都被标记* **for**(Cell[] cellList: **cells**){**for**(Cell cell:cellList){*//对于每一个单元格* **if**(!(cell.**isMined**||cell.**isRevealed**)){*//不满足获胜条件1* **for**(Cell[] cellList2: **cells**){**for**(Cell cell2:cellList2){*//对于每一个单元格* **if**(cell2.**isMined**&&!cell2.**isFlagged**){*//不满足获胜条件2* **return false**;  
 }  
 }}  
 **return true**;*//满足获胜条件2* }  
 }}  
 **return true**;*//满足获胜条件1*}

以上是判断获胜的代码；

**private class** CellMouseListener **extends** MouseAdapter{  
 **public void** mouseClicked(MouseEvent e){  
 *// 获得触发此次鼠标事件的Cell对象* Cell sourceCell = (Cell)e.getSource();  
 *// 获得鼠标事件的类型，MouseEvent.BUTTON1为单击鼠标左键* **if**(e.getButton() == MouseEvent.***BUTTON1***) {  
 sourceCell.**isFlagged**=**false**;*//单击左键时需移除旗子  
 // 如果当前Cell对象里面有地雷，则游戏结束；否则对该Cell对象执行挖雷操作* **if** (sourceCell.**isMined**) {  
 **frame**.**menu**.**timer**.stop();  
 JOptionPane.*showMessageDialog*(**frame**,**"Game over!"**,**"Game over!"**,JOptionPane.***INFORMATION\_MESSAGE***,**new** ImageIcon(**"assets/mine.png"**));  
 sourceCell.setIcon(**new** ImageIcon(**"assets/mine.png"**));  
 } **else** {  
 revealCell(sourceCell.**row**, sourceCell.**col**);  
 }  
 }  
 **else if**(e.getButton() == MouseEvent.***BUTTON3***){ *//MouseEvent.BUTTON3为单击鼠标右键  
 // 果该Cell对象上插了旗子，那么就去掉旗子；否则插上旗子。* **if**(sourceCell.**isFlagged** || sourceCell.**isRevealed**){*//如果该Cell对象已经被挖雷，不应插上旗子* sourceCell.**isFlagged**=**false**;  
 sourceCell.paint();  
 }  
 **else**{  
 sourceCell.**isFlagged**=**true**;  
 sourceCell.paint();  
 }  
 }  
 **frame**.**menu**.refreshBar();  
 **if**(hasWon()){  
 **frame**.**menu**.**timer**.stop();  
 JOptionPane.*showMessageDialog*(**frame**,**"You win! Time:"**+**frame**.**menu**.**timeBar**.getText(),**"You win!"**,JOptionPane.***PLAIN\_MESSAGE***);  
 }  
 }  
}

以上是监听每个块被点击的内部类CellMouseListener的主要代码，它继承了MouseAdapter类，重写了mouseClicked方法。方法中对用户所做的左键或右键操作分别处理，是整个扫雷程序的重要组成部分。

# 运行结果展示



可详见视频演示。

# 总结和收获

在此次实践中，我基本实现了一个功能完好的扫雷游戏。通过对本项目的实现，我对javax.swing的使用有了初步认识、练习了Java语言的运用，也加深了对面向对象程序设计的理解。实践中，扫雷游戏的每一部分都可以抽象为一个对象，通过继承javax.swing的不同类，实现组件的多样化和组合。实践中接触了JButton,JPanel,JMenu,JMenuItem,JTextField等组件类型，而它们都实现了JComponent，即都是组件的一种，这就是继承与多态的具体体现。

目前，扫雷项目虽然运行基本无误，但仍有一些需要改进的地方，例如，目前计时器虽然调用了stop方法但仍不停止，未能发现其原因；还有，项目中用了两种监听器，一种是实现ActionListener，一种是继承MouseAdapter，且一个是内部类而一个是类直接实现接口，为便于理解、查看和后续功能的添加，可以考虑将它们统一形式或合并为一个监听器。

以下是附加部分的分值表填写：

|  |  |
| --- | --- |
| 功能列表 | 分值 |
| 扩展1 | 7 |
| 扩展2 | 7 |
| 扩展3 | 4 |
| 填表人：张子杰 | |

# 源代码

### MineSweeperMain.java

**package** minesweeper;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.io.Serial;  
  
  
*/\*\*  
 \* 扫雷游戏的主程序。  
 \* 定义了扫雷游戏的主面板，其中包括了菜单面板、游戏面板和新游戏面板。  
 \*/***public class** MineSweeperMain **extends** JFrame {  
 @Serial  
 **private static final long *serialVersionUID*** = 1L;  
 *//为每个面板都传入主面板，以便它们对主面板或者主面板中的其它面板进行操作* **public** GameBoardPanel **board** = **new** GameBoardPanel(**this**);  
 **public** NewGamePanel **newGame** = **new** NewGamePanel(**this**);  
 **public** GameMenuPanel **menu** = **new** GameMenuPanel(**this**);  
  
 **public** MineSweeperMain(){  
 Container cp = **this**.getContentPane();  
 cp.setLayout(**new** BorderLayout());  
 **board**.newGame(NewGamePanel.***EASY***);  
 cp.add(**menu**,BorderLayout.***NORTH***);  
 cp.add(**board**, BorderLayout.***CENTER***);  
 cp.add(**newGame**, BorderLayout.***SOUTH***);  
 pack(); *// Pack the UI components, instead of setSize()* setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 setTitle(**"Minesweeper"**);  
 setVisible(**true**);  
 }  
 **public static void** main(String[] args){  
 *// 使用安全的方式启动下面的构造函数* SwingUtilities.*invokeLater*(MineSweeperMain::**new**);  
 }  
}

### GameMenuPanel.java

**package** minesweeper;  
  
**import** javax.swing.\*;  
**import** java.awt.\*;  
*/\*\*  
 \* 这个类是扫雷游戏上方的菜单栏、进度条和计时器部分。  
 \* 菜单栏中有一个File菜单，它又包含三个菜单项。  
 \* 进度条和计时器均使用JTextField实现，并被添加到一个JPanel中。  
 \*/***public class** GameMenuPanel **extends** JPanel {  
 **private** JMenuBar **menuBar** = **new** JMenuBar();*//菜单栏* **private** JMenu **menuFile** = **new** JMenu(**"File"**);*//一级菜单* **public** JMenuItem **menuItemNewGame** = **new** JMenuItem(**"New Game"**);  
 **private** JMenuItem **menuItemResetGame** = **new** JMenuItem(**"Reset Game"**);  
 **private** JMenuItem **menuItemExit** = **new** JMenuItem(**"Exit"**);  
 **private** JPanel **progressAndTimePanel** = **new** JPanel();  
 **private** JTextField **progressBar** = **new** JTextField();  
 **public** JTextField **timeBar** = **new** JTextField(**"00:00"**);  
 **public** Timer **timer**;  
 **private long timeStart**;  
 **private** MineSweeperMain **frame**;*//扫雷的主面板* **public** GameMenuPanel(MineSweeperMain frame){  
 **this**.**frame**=frame;  
 setLayout(**new** BorderLayout());  
 *//添加菜单栏、添加菜单、添加菜单项、为菜单项添加监听器* add(**menuBar**, BorderLayout.***CENTER***);  
 **menuBar**.add(**menuFile**);  
 **menuFile**.add(**menuItemNewGame**);  
 **menuFile**.add(**menuItemResetGame**);  
 **menuFile**.add(**menuItemExit**);  
 **menuItemNewGame**.addActionListener(frame.**newGame**);  
 **menuItemResetGame**.addActionListener(e -> frame.**board**.resetGame());  
 **menuItemExit**.addActionListener(e -> frame.dispose());  
 add(**progressAndTimePanel**,BorderLayout.***SOUTH***);  
 *//添加进度条和计时栏、给该栏添加进度条、添加计时器* **progressAndTimePanel**.setLayout(**new** BorderLayout());  
 **progressAndTimePanel**.add(**progressBar**,BorderLayout.***WEST***);  
 **progressAndTimePanel**.add(**timeBar**,BorderLayout.***EAST***);  
 }  
  
 **public void** refreshBar() {  
 **int** notFoundMines=**frame**.**board**.**numMines**;  
 **for**(**int** row = 0; row < GameBoardPanel.*ROWS*; ++row){  
 **for**(**int** col = 0; col < GameBoardPanel.*COLS*; ++col){  
 **if**(**frame**.**board**.**cells**[row][col].**isFlagged**){  
 notFoundMines--;  
 }  
 }  
 }  
 System.***out***.print(notFoundMines);  
 String text = **"剩余地雷数："** + notFoundMines + **"/"** + **frame**.**board**.**numMines**;  
 **if**(notFoundMines>0){  
 text+=**"⬛"**.repeat(notFoundMines);  
 }  
 progressBar.setText(text);  
 }  
 **public void** startClock(){  
 timeBar.setText(**"00:00"**);  
 timeStart=System.currentTimeMillis();  
 timer = **new** Timer(1000, e -> {  
 **long** timePeriod=System.currentTimeMillis()-timeStart;  
 timeBar.setText(String.format(**"%02d:%02d"**,timePeriod/60000,(timePeriod/1000)%60));  
 });  
 timer.start();  
 }  
}

### NewGamePanel.java

**package** minesweeper;  
  
**import** javax.swing.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
*/\*\*  
 \* 这个类是扫雷游戏下方的开始游戏区域。  
 \* 同时，这个类实现了ActionListener接口，可以对开始新游戏的行为进行监听处理。  
 \* 单击开始游戏后，该区域将显示Easy,Intermediate,Difficult三种难度供玩家选择。  
 \* 玩家选择后以对应难度开始游戏。  
 \*/***public class** NewGamePanel **extends** JPanel **implements** ActionListener {  
 **private** JButton **btnNewGame** = **new** JButton(**"New Game"**);  
 **private** JButton **btnEasy** = **new** JButton(**"Easy"**);  
 **private** JButton **btnIntermediate** = **new** JButton(**"Intermediate"**);  
 **private** JButton **btnDifficult** = **new** JButton(**"Difficult"**);  
 **private** MineSweeperMain **frame**;*//当前的游戏主面板* **public static final int *EASY*** = 1;  
 **public static final int *INTERMEDIATE*** = 2;  
 **public static final int *DIFFICULT*** = 3;  
 **public** NewGamePanel(MineSweeperMain frame){  
 **this**.**frame**=frame;  
 add(**btnNewGame**);  
 *//为按钮添加鼠标监听器，即新游戏面板自身* **btnNewGame**.addActionListener(**this**);  
 **btnEasy**.addActionListener(**this**);  
 **btnIntermediate**.addActionListener(**this**);  
 **btnDifficult**.addActionListener(**this**);  
 }  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 removeAll();  
 **if**(e.getSource()==**btnNewGame**||e.getSource()==**frame**.**menu**.**menuItemNewGame**){*//单击新游戏按钮时，让用户选择游戏难度* add(**btnEasy**);  
 add(**btnIntermediate**);  
 add(**btnDifficult**);  
 updateUI();  
 }  
 **else**{*//单击难度按钮时，以对应难度开始游戏* add(**btnNewGame**);  
 updateUI();  
 **if**(e.getSource()==**btnEasy**){  
 **frame**.**board**.newGame(***EASY***);  
 }  
 **if**(e.getSource()==**btnIntermediate**){  
 **frame**.**board**.newGame(***INTERMEDIATE***);  
 }  
 **if**(e.getSource()==**btnDifficult**){  
 **frame**.**board**.newGame(***DIFFICULT***);  
 }  
 }  
 }  
}

### GameBoardPanel.java

**package** minesweeper;  
**import** java.awt.\*;  
**import** javax.swing.\*;  
**import** java.awt.event.\*;  
**import** java.io.Serial;  
  
*/\*\*  
 \* 这是扫雷游戏的游戏面板，即雷区。  
 \* 单击鼠标左键对单元格执行挖雷操作。  
 \* 单击鼠标右键用来对单元格执行添加标记，或者移除标记，标记疑似有地雷的单元格。  
 \* 如果所有没有地雷的单元格都执行了挖雷操作，或玩家标记了所有地雷，那么玩家赢得游戏。  
 \* 如果对某个有地雷的单元格执行了挖雷操作，那么玩家输。  
 \*/***public class** GameBoardPanel **extends** JPanel {  
 @Serial  
 **private static final long *serialVersionUID*** = 1L;  
 *// 设定了“棋盘”中有多少个Cell对象* **public static int** *ROWS*;  
 **public static int** *COLS*;  
 *// 每一个Cell对象的尺寸大小，依次大小再去计算“棋盘”的大小* **public static int** *CELL\_SIZE* = 35;  
 **private** MineSweeperMain **frame**;  
 **public** Cell[][] **cells**;  
 **int numMines** = 10;  
 **public** GameBoardPanel(MineSweeperMain frame){**this**.**frame**=frame;}  
 **public void** newGame(**int** difficulty){  
 **switch** (difficulty) {*//根据难度调整大小和地雷数量* **case** (NewGamePanel.***EASY***) -> {  
 *ROWS* = 10;  
 *COLS* = 10;  
 **numMines** = 10;  
 }  
 **case** (NewGamePanel.***INTERMEDIATE***) -> {  
 *ROWS* = 15;  
 *COLS* = 15;  
 **numMines** = 20;  
 }  
 **case** (NewGamePanel.***DIFFICULT***) -> {  
 *ROWS* = 20;  
 *COLS* = 20;  
 **numMines** = 30;  
 }  
 }  
 **cells**=**new** Cell[*ROWS*][*COLS*];  
 **super**.setLayout(**new** GridLayout(*ROWS*, *COLS*, 1, 1));  
 CellMouseListener listener = **new** CellMouseListener();*//为所有的Cell单元对象创建一个共享的鼠标事件监听器* MineMap mineMap = **new** MineMap(*ROWS*,*COLS*,**numMines**);*//通过MineMap获得新游戏中的地雷数据的布局* **super**.removeAll();*//清空上次游戏里的Cell* **for**(**int** row = 0; row < *ROWS*; ++row){  
 **for**(**int** col = 0; col < *COLS*; ++col){  
 **cells**[row][col] = **new** Cell(row, col);  
 **cells**[row][col].addMouseListener(listener);  
 **cells**[row][col].newCell(mineMap.**isMined**[row][col]);  
 **super**.add(**cells**[row][col]);  
 }  
 }  
 *//根据* **super**.setPreferredSize(**new** Dimension(*CELL\_SIZE* \* *COLS*, *CELL\_SIZE* \* *ROWS*));  
 **frame**.**menu**.refreshBar();  
 **frame**.**menu**.startClock();  
 **frame**.pack();  
 }  
 **public void** resetGame(){  
 **for**(**int** row = 0; row < *ROWS*; ++row){  
 **for**(**int** col = 0; col < *COLS*; ++col){  
 **cells**[row][col].**isRevealed**=**false**;  
 **cells**[row][col].**isFlagged**=**false**;  
 **cells**[row][col].paint();  
 }  
 }  
 **frame**.**menu**.refreshBar();  
 **frame**.**menu**.startClock();  
 }  
 *// 获得[srcRow, srcCol]Cell单元对象周围的8个邻居的地雷总数* **private int** getSurroundingMines(**int** srcRow, **int** srcCol){  
 **int** numMines = 0;  
 **for**(**int** row = srcRow-1; row <= srcRow+1; ++row){  
 **for**(**int** col = srcCol-1; col <= srcCol+1; ++col){  
 **if**(row >= 0 && row < *ROWS* && col >= 0 && col < *COLS*)  
 **if**(**cells**[row][col].**isMined**) numMines++;  
 }  
 }  
 **return** numMines;  
 }  
 *// 对[srcRow, srcCol]Cell单元对象执行挖雷操作  
 // 如果该单元格对象中的标记的雷的数量为0，那么就自动递归对其周围8个邻居执行挖雷操作* **private void** revealCell(**int** srcRow, **int** srcCol){  
 **int** numMines = getSurroundingMines(srcRow, srcCol);  
 **cells**[srcRow][srcCol].setText(numMines + **""**);  
 **cells**[srcRow][srcCol].**isRevealed** = **true**;  
 **cells**[srcRow][srcCol].**isFlagged** = **false**;  
 **cells**[srcRow][srcCol].paint();  
 **if**(numMines == 0){  
 **for**(**int** row = srcRow-1; row <= srcRow+1; ++row){  
 **for**(**int** col = srcCol-1; col <= srcCol+1; ++col){  
 **if**(row >= 0 && row < *ROWS* && col >= 0 && col < *COLS*)  
 **if**(!**cells**[row][col].**isRevealed**) revealCell(row, col);  
 }  
 }  
 }  
 }  
 *// 用来判断玩家是否已经赢得此次游戏* **public boolean** hasWon(){  
 *// 获胜条件1：所有单元格都成功执行了挖雷操作  
 // 获胜条件2：所有的地雷都被标记* **for**(Cell[] cellList: **cells**){**for**(Cell cell:cellList){*//对于每一个单元格* **if**(!(cell.**isMined**||cell.**isRevealed**)){*//不满足获胜条件1* **for**(Cell[] cellList2: **cells**){**for**(Cell cell2:cellList2){*//对于每一个单元格* **if**(cell2.**isMined**&&!cell2.**isFlagged**){*//不满足获胜条件2* **return false**;  
 }  
 }}  
 **return true**;*//满足获胜条件2* }  
 }}  
 **return true**;*//满足获胜条件1* }  
 *// 定义一个内部类，该类的作用为鼠标事件监听器* **private class** CellMouseListener **extends** MouseAdapter{  
 **public void** mouseClicked(MouseEvent e){  
 *// 获得触发此次鼠标事件的Cell对象* Cell sourceCell = (Cell)e.getSource();  
 *// 获得鼠标事件的类型，MouseEvent.BUTTON1为单击鼠标左键* **if**(e.getButton() == MouseEvent.***BUTTON1***) {  
 sourceCell.**isFlagged**=**false**;*//单击左键时需移除旗子  
 // 如果当前Cell对象里面有地雷，则游戏结束；否则对该Cell对象执行挖雷操作* **if** (sourceCell.**isMined**) {  
 JOptionPane.*showMessageDialog*(**frame**,**"Game over!"**,**"Game over!"**,JOptionPane.***INFORMATION\_MESSAGE***,**new** ImageIcon(**"assets/mine.png"**));  
 sourceCell.setIcon(**new** ImageIcon(**"assets/mine.png"**));  
 **frame**.**menu**.**timer**.stop();  
 } **else** {  
 revealCell(sourceCell.**row**, sourceCell.**col**);  
 }  
 }  
 **else if**(e.getButton() == MouseEvent.***BUTTON3***){ *//MouseEvent.BUTTON3为单击鼠标右键  
 // 果该Cell对象上插了旗子，那么就去掉旗子；否则插上旗子。* **if**(sourceCell.**isFlagged** || sourceCell.**isRevealed**){*//如果该Cell对象已经被挖雷，不应插上旗子* sourceCell.**isFlagged**=**false**;  
 sourceCell.paint();  
 }  
 **else**{  
 sourceCell.**isFlagged**=**true**;  
 sourceCell.paint();  
 }  
 }  
 **frame**.**menu**.refreshBar();  
 **if**(hasWon()){  
 JOptionPane.*showMessageDialog*(**frame**,**"You win! Time:"**+**frame**.**menu**.**timeBar**.getText(),**"You win!"**,JOptionPane.***PLAIN\_MESSAGE***);  
 **frame**.**menu**.**timer**.stop();  
 }  
 }  
 }  
}

### Cell.java

**package** minesweeper;  
**import** java.awt.Color;  
**import** java.awt.Font;  
**import** java.io.Serial;  
**import** javax.swing.\*;  
**import** javax.swing.border.EmptyBorder;  
  
*/\*\*  
 \* 类Cell是对JButton的类定制（也就是JButton的一个子类，其目的是表示扫雷游戏的一个单元格；  
 \* 该类中定义了row/column属性以及相关状态函数。  
 \*/***public class** Cell **extends** JButton{  
 *// 为Cell单元格定义若干颜色和字体常量  
 // 这些常量将随着Cell单元格的状态变化而被使用* @Serial  
 **private static final long *serialVersionUID*** = 1L; *// to prevent serial warning* **public static final** Color ***BG\_NOT\_REVEALED*** = Color.***CYAN***;  
 **public static final** Color ***FG\_NOT\_REVEALED*** = Color.***RED***; *// flag, mines* **public static final** Color ***BG\_REVEALED*** = Color.***DARK\_GRAY***;  
 **public static final** Color ***FG\_REVEALED*** = Color.***YELLOW***; *// number of mines* **public static final** Font ***FONT\_NUMBERS*** = **new** Font(**"Monospaced"**, Font.***BOLD***, 20);  
  
 *// 定义Cell对象的属性，比如row和col值用来表示单元格在最终“棋盘”上的位置定位* **public int row**, **col**;  
 **public boolean isRevealed**; *// 标记是否已经被挖出？* **public boolean isMined**; *// 标记是否是地雷？* **public boolean isFlagged**; *// 标记是否被玩家插上了一个小红旗* **public** Cell(**int** row, **int** col){  
 **this**.**row** = row;  
 **this**.**col** = col;  
 **super**.setFont(***FONT\_NUMBERS***);*//设置字体* **super**.setBorder(**new** EmptyBorder(5,5,5,5));*//设置边距* }  
  
 **public void** newCell(**boolean** isMined){  
 **this**.**isRevealed** = **false**;  
 **this**.**isFlagged** = **false**;  
 **this**.**isMined** = isMined;  
 **super**.setEnabled(**true**);  
 **super**.setText(**""**);  
 **super**.setIcon(**null**);  
 paint();  
 }  
 */\*\* 基于单元格的状态进行绘制 \*/* **public void** paint(){  
 **super**.setForeground(**isRevealed**? ***FG\_REVEALED***: ***FG\_NOT\_REVEALED***);  
 **super**.setBackground(**isRevealed**? ***BG\_REVEALED***: ***BG\_NOT\_REVEALED***);  
 **if**(**isFlagged**){  
 setIcon(**new** ImageIcon(**"assets/flag.png"**));  
 }  
 **else**{  
 setIcon(**null**);  
 }  
 **if**(!**isRevealed**){  
 setText(**""**);  
 }  
 }  
}

### MineMap.java

**package** minesweeper;  
**import** java.util.Random;  
  
*/\*\* 这个类主要用来存储地雷在单元格中的位置， \*/***public class** MineMap {  
 **boolean**[][] **isMined**;  
 **public** MineMap(**int** rows,**int** cols,**int** numMines){  
 **this**.**isMined** = **new boolean**[rows][cols];  
 Random randomNumber = **new** Random();  
 **for**(**int** i=0;i<numMines;i++){  
 **int** x=randomNumber.nextInt(0,GameBoardPanel.*ROWS*);  
 **int** y=randomNumber.nextInt(0,GameBoardPanel.*COLS*);*//随机生成一个放置地雷的位置* **if**(!**isMined**[x][y]) **isMined**[x][y]=**true**;  
 **else** i--;*//若该位置已有地雷，此次随机作废* }  
 }  
}